Towards Caching Strategies for Quantum Simulators

Immanuel Trummer
itrummer@cornell.edu
Cornell University
Ithaca, NY, USA

ABSTRACT

This paper proposes to use caching strategies to speed up quantum simulation by avoiding redundant work across multiple simulator runs.

VLDB Workshop Reference Format:
Immanuel Trummer. Towards Caching Strategies for Quantum Simulators. VLDB 2026 Workshop: QCDKM 2026 - 2nd Workshop on Quantum Computing and Data/Knowledge Management.

1 INTRODUCTION

Simulating quantum computers on classical hardware is necessary, for instance, to guide the design of the next generation of quantum hardware. Quantum simulation is computationally expensive, a problem that is exacerbated by the growing number of qubits on current quantum accelerators. This makes it crucial to reduce computational overheads for quantum simulation wherever possible. Most prior work on quantum simulation considers different simulator runs in isolation. However, this perspective neglects the potential to reuse partial results across invocations. This abstract proposes to adapt caching strategies, used by relational database systems for decades, to avoid redundant computation in quantum simulation as well.

2 PROBLEM SETTING

This abstract refers to a scenario in which a quantum simulator simulates the same quantum circuit repeatedly. In this scenario, caching can help to reduce computational overheads of quantum simulation by avoiding redundant work across simulations. While consecutive simulator runs focus on the same circuit, it is assumed that the input parameters used for the circuit may change from one run to the next. This corresponds to users testing a quantum algorithm on different problem instances. In an extension of the aforementioned scenario, we can assume that not one but multiple quantum circuits should be simulated repeatedly. If these circuits have common parts, cache content may be reused across different circuits. This becomes particularly attractive in scenarios in which users apply alternative circuits to the same problem instances, e.g., to compare the performance of different quantum algorithms. In that case, intermediate results concerning components that are shared by different circuits can be directly reused.

Proceedings of the VLDB Endowment. ISSN 2150-8097.

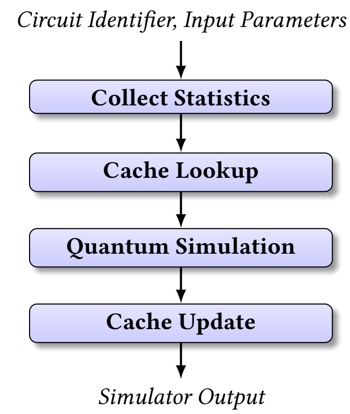

Figure 1: Overview of cache-aware quantum simulation.

3 SYSTEM OVERVIEW

Figure 1 shows how to integrate caching strategies into quantum simulators. Users submit the identifier of a quantum circuit, together with input parameters. To guide future caching decisions, the cache-aware quantum simulator must collect statistics regarding the input parameters (allowing it to calculate the reuse probability for certain cache entries). Based on the target circuit and input parameters, the cache-aware quantum simulator performs a cache lookup, identifying cache entries that can help to speed up the current quantum simulation. After that, the quantum simulator performs remaining work, starting from cached quantum states wherever possible. The cache-aware quantum simulator then applies a cache replacement policy to determine which, if any, of the intermediate results generated during the current simulation to store in the cache. The final output is returned to the user.

4 CACHE ENTRIES

Different quantum simulators represent quantum states in different formats. For instance, quantum simulators based on Schrödinger's algorithm [1, 2] represent quantum states as a vector, associating each combination of qubit values with an amplitude. Compression may be used to reduce the size of that vector, compared to a naive representation that takes space in $O(2^k)$ (where k is the number of qubits). From the perspective of the caching component, the precise representation does not matter, as long as meta-data such as the representation size and the cost savings via reuse are known.

5 CACHING POLICY

The total space that can be used for caching is limited. This motivates a cache replacement policy that weighs the benefits of caching different quantum states. First, the caching policy should take into account the probability that a cache entry can be reused, $P_{use}(e)$ for a cache entry e. In the simplest version, this probability is based on statistics collected about the input workload, using the ratio of inputs for which the cache entry could be used. A more sophisticated version takes into account other cache content. E.g., caching quantum states that appear later in a circuit can render quantum states that appear earlier in the circuit redundant (for the same input). Second, the caching policy should take into account $C(e)$ the overheads of re-computation if cache entry e is not available (preferring to cache entries where recomputation is expensive). In the simplest version, cost can be approximated by the cost of computing the stored entry. A more sophisticated cost metric takes into account that the cache entry may be recomputed in a different way, exploiting other cache entries as a starting point. Finally, the expected benefit and the storage $S(e)$ required for cache entry e can be integrated using a formula such as $P_{use} \cdot C(e)/S(e)$, replacing existing entries if their value is lower than for the new entry. Future research must evaluate caching strategies with different degrees of sophistication when comparing candidate cache entries in terms of their empirical performance.

REFERENCES

[1] Yasunari Suzuki, Yoshiaki Kawase, Yuya Masumura, Yuria Hiraga, Masahiro Nakadai, Jiabao Chen, Ken M Nakanishi, Kosuke Mitarai, Ryosuke Imai, Shiro Tamiya, Takahiro Yamamoto, Tennin Yan, Toru Kawakubo, Yuya O Nakagawa, Yohei Ibe, Youyuan Zhang, Hirotsugu Yamashita, Hikaru Yoshimura, Akihiro Hayashi, and Keisuke Fujii. 2021. Qulacs: A fast and versatile quantum circuit simulator for research purpose. *Quantum* 5 (2021). https://doi.org/10.22331/Q-2021-10-06-559 arXiv:2011.13524

[2] Xin Chuan Wu, Sheng Di, Emma Maitreyee Dasgupta, Franck Cappello, Hal Finkel, Yuri Alexeev, and Frederic T Chong. 2019. Full-state quantum circuit simulation by using data compression. In *International Conference for High Performance Computing, Networking, Storage and Analysis, SC.* https://doi.org/10.1145/3295500.3356155 arXiv:1911.04034